International Journal of Intelligent Information Technologies
Volume 18 • Issue 3

SDN-Based Traffic Monitoring in Data Center Network Using Floodlight Controller

Himanshu Sahu, University of Petroleum and Energy Studies, India
https://orcid.org/0000-0003-1202-6286
Rajeev Tiwari, University of Petroleum and Energy Studies, India
Sumit Kumar, University of Petroleum and Energy Studies, India*

ABSTRACT

Data center networks are the backbone of IT infrastructure and cloud services. According to traffic pattern research, a small group of flows transport the vast majority of the bytes and are referred to as elephant flows. Proper management of such traffic flows can enhance overall performance and energy efficiency. Software-defined network (SDN) is a fresh networking model that provides a centralized control plane (i.e., controller). The controller can be utilized for traffic monitoring by collecting the network flows at the controller. In this research, a new mechanism has been provided to detect such flows, which requires continuous polling of all switches. The proposed method depends on passive querying so it does not require additional traffic. The result shows the successful detection of elephant flow and cheetah flow that can be rerouted to improve the quality of service (QoS).

KEYWORDS

Cheetah Flow, Data Center Network, Elephant Flow, Floodlight Controller, Flow Rate, Mice Flow, SDN, SDN Controller, Traffic Engineering

INTRODUCTION

SDN (Sahu & Hungyo, 2018; Singh et al., 2019) is an emerging network that is capable of transforming the traditional network architecture due to its support for programmable networks and open protocols as discussed in (Nunes et al., 2014). The SDN paradigm bifurcates the traditional network architecture with distributed forwarding data plane and a centralized control plane. The network's "brain" is the SDN Controller which is used to communicate with the hardware architecture and controls traffic on the network. The custom applications are developed in the SDN controller to support a variety of responsibilities like load balancing; anomaly detection, and dynamic centralized routing decision.

The data center is the backbone of IT infrastructure and consumer service-based applications. The exponential growth of mobile computing and the application-based environment caused the data

DOI: 10.4018/IJIIT.309590
*Corresponding Author

centers to carry a huge amount of traffic with unpredictable behavior. As per the work presented by Index, (2016); Kumar et al., (2021), *"The amount of annual global data center traffic in 2015 is already estimated to be 4.7 ZB and by 2020 will triple to reach 15.3 ZB per year"*. Therefore, Data Center requires better traffic management for efficient and optimized resource utilization.

Traffic Engineering (TE) consists of methods devised to optimize the performance of the data network. TE uses static or dynamic analysis of traffic for better management to avoid congestion and better utilization of bandwidth (Kumar & Tiwari, 2021). Elephant flows are presented by Kandula et al. (2009), which came in less frequency but consume huge bandwidth. This causes sudden network congestion. The detection of such flows can increase the efficiency of the overall system since the post-detection the flows be rerouted to load balance the network.

The Datacenter (DC) traffic is of two types, user traffic, and server traffic. User traffic consists of data by the services provided to the user containing external traffic. Whereas, the server traffic is generated due to the inter-server communication. The DC network requires significant aggregate bandwidth. Typically, it has a tree-like architecture made up of routing and switching components. In this kind of implementation, only a small portion of the total bandwidth is made available to the edge network.. Therefore, a topology such as the Fat-tree topology provides better distribution of bandwidth and ensures network reliability.

A comprehensive view and control of the entire network are provided by SDN. It has been successfully implemented in different DC networks like Google B4 (Jain et al., 2013). SDN provides network programmability and dynamic policy updates. SDN-based traffic engineering solutions are effective in DC networks as discussed by Ian et al. (2014). In the present paper, a method is provided for elephant flow that is directly implemented as a module in the floodlight controller, which is further elaborated in the paper. The proposed method requires polling from the network switches and identifies the elephant flow and the cheetah flows in the network. The proposed method uses a dynamically computed threshold value for flow detection and its value is computed based on the network traffic characteristics.

The rest of the paper is organized as follows. The details of the recent works suggested by various researchers are discussed in the "Background" section. Subsequent section, after background the section, provides the "Proposed Methodology". The result analysis has been discussed in "Results" section and finally, the paper is concluded in the "Conclusion and Future Works" section.

BACKGROUND

Traffic Engineering in SDN Network

Traffic Engineering (TE) is the pool of practices for performance optimization of data communication networks that includes traffic pattern analysis, forecast, and handling traffic activities. The centralized SDN controller-based TE solution can help in achieving quick adaptation and dynamic updates of network policy. Furthermore, it is widely used for continuous monitoring of network performance parameters such as link utilization, throughput, delay, etc.

The network control plane and the data forwarding plane are separated by the SDN system. It is used to provide the consolidated vision of distributed network instances. The TE handles several aspects of the SDN network as discussed by Ian et al. (2014) which are broadly categorized as flow management, fault tolerance, topology update, traffic analysis, and characterization. The present work is coming under the fourth category, which will be discussed in detail along with the rest of the category.

Flow Management

In an SDN network, switches are mere forwarding devices working under the guidance of the controller. The controller sets up the flow, which is followed by the switches. Flow management

can be considered as the global routing decision, which can be used for packet routing. Switch load balancing in a network where multiple equal-cost paths are available, the switch can load balance the traffic equally to each path. The hash-based technique can be used for load balancing. This method has a few limitations. Two heavy loads may flow and get hashed on the same link. The flow rule can be installed based on the load on the particular link known as load-sensitive forwarding.

Towards this, the Hedera system has been suggested by Al-Fares et. al. (2010) for scalability and adaptive scheduling of network flows. The Hedera method adaptively schedules a multi-stage switching for effective utilization of available network resources. At a particular frequency, the system polls the switch to detect the elephant flow and after a specific threshold value, it recalculates the path and redirects that flow. It detects the elephant flow at the edge switches.

Topology Update

Topology updates are known as the planned changes in the network policy. Methods such as time-based configuration change send the configuration updates to switches. Based on this, switches update their flow table entries.

SDN Traffic Analysis

Monitoring is necessary for network management, which requires timely and accurate statistics of the traffic. Statistics are available at the controller, switches, or on the end host. Statistics collection leads to excessive overhead on the controller and switches link so there is a need to find an effective solution for the statistics collection.

The PayLess system is discussed by Chowdhury et al. (2014), which is developed as a network analysis framework for SDN. The PayLess offers an abstract representation of the network and a way to examine the resource availability information. It query-based system that adaptively changes the polling frequency so that the load of statistics collection would be reduced. It is created as a pluggable element with well-defined interfaces for increased connectivity with other frameworks. OpenTM (Tootoonchian et al., 2010) is also a query-based system but instead of polling all the switches in the path, it selectively polls only one switch at a time. It is a traffic characteristic approximation system for the OpenFlow networks. The system uses the implicit features of OpenFlow switches for analyzing the traffic properties with low overhead. The OpenTM examines the routing data from the controller to dynamically select the appropriate switches to retrieve flow statistics. The results of the OpenTM show that the coverage within 10 queries is significantly faster than the existing traffic matric approximation strategies. The FlowSense (Yu et al., 2013) is a passive push-based method, which is more accurate and has lesser overhead in the system. In this scheme, the control messages are forwarded from switches towards the centralized controller to approximate the performance. The scheme computes the utilization of communication channels between the switches. MicroTE (Benson et al., 2011) is a tool for the DC with a monitoring component in the server. The system adapts to the variations in the traffic by using the short-term and partial prediction of the traffic matrix. It allows triggered updates to be sent to the controller. The technique provided by Queiroz et al. (2019) is a fine-grained big data-based technique to provide more detailed traffic patterns, which can enhance the effectiveness of the TE solutions. Table 1 provides the summary of the recent work done in the application of SDN in traffic engineering.

Elephant Flow Detection Models

In the DC network, traffic control is required to efficiently use the bisection bandwidth and it is offered using the Fat-tree topology. A huge quantity of data-carrying elephant flows must be promptly identified and managed. In this field, the work that has been done so far are described as follows.

One approach is that the application sending data itself identifies the flows as elephant flow. Such applications need either machine learning or simple packet header matching. Due to the enormous amount of traffic in DC, this approach is not useful.

Table 1. Recent Work related to traffic engineering using SDN

S.No.	Paper	Approach	Merits/Demerits
1	OpenTM (Tootoonchian et al. 2010)	Host to traffic flow estimator based on Open flow	1. It uses an intelligent mechanism for the selection of those switches which are involved in polling by using the routing information. 2. It accurately calculates the traffic flow with low overhead.
2	FlowSense (Yu et al., 2013)	Flow sense is designed to provide the bandwidth utilization monitoring solution for Flow-based networks such as the OpenFlow network. This utilizes the PacketIN and FlowRemoved message to find the link utilization.	1. The system uses the passive approach for switch polling. 2. Since the method is based on asynchronous events, continuous monitoring is not available. 3. Traffic patterns also affect the performance of the system i.e. if a lot of tortoise flow is present the monitoring output will available with high latency. Whereas, if a lot of rat flow is present, then the monitoring traffic will be high.
3	(Lin et al., 2014)	Proposed hierarchical way of pulling statistics from the switches to avoid unrelated passive pulling.	By using the HSP mechanism, the querying traffic is reduced.
4	(Luong et al., 2016)	Designed the Throughput Monitor module for link traffic of each switch. Proposed a new method for packet forwarding.	1. In this mechanism, any application can access the data collected by the monitor module.
5	(Xing et al., 2016)	Create a two-stage flow detection method. Initial phase simply fetches and samples large flow and in the second phase, the controller verifies if exactly should be considered an elephant flow.	1. The mechanism attempts to remove the problem of limited TCAM storage at the controller. 2. It distributes the load of elephant flow detection to all switches.
6	(Queiroz et al., 2019)	Provided big data-based approach for TE solution for SDN networks	1. The method considers the switch's counter values as streaming data. 2. More fine-grained information, which provides switch-wise as well as aggregate data.
7	(Hamdan et al., 2020)	Provided a flow-aware elephant detection method by adaptively changing the threshold of the elephant flow.	1. Proposed two classifiers one for switches and the other for the controller. 2. The mice flow can be filtered on switches so that the traffic is reduced and sent to switches for classification. 3. It outperforms the existing methods

The second approach is to analyse every flow on first switch. The Pull based approach is used to get the statistics from the switch by polling each switch at regular intervals and the statistics are used for the detection of elephant flow. The works suggested in Hedera (Al-Fares et al., 2010) and by Lin et al. (2014) are examples that are using this approach. This method uses a lot of switch resources, hence it does not scale well to a big network and requires high bandwidth consumption between the controller and switch. It will cause a performance bottleneck to the system.

The third approach is based on sampling, in which using sampling features like sflow or netflow, a controller takes samples of packets from all of the switches' ports (Zaw & Maw et al., 2019; Sarvotham et al., 2001). The method just transfers the packet header to the controller and samples only a small portion of the packets (usually 1 in 1000) at the switches. This approach is incapable of dependably detecting an elephant flow until carried more than 10K packets. This approach also suffers from the additional overhead of switching to controller traffic.

The other approach, Mahout (Curtis et al., 2011) is based on the detection of elephant flow on end-hosts and uses the in-band signalling methods to mark the flow as elephant flow, which is

processed by the controller. The Mahout system has been presented for detection of the elephant flow at the end-hosts. For this, the system considers a shim layer instead of implementing the monitoring procedure in the network switches. In the Mahout system, the traffic which is not detected as the elephant flow has been forwarded using a static load balancing strategy. However, the scheme monitors and manages the elephant flows only.

Freeway (Wang et al., 2014) has proposed the detection of not only large-sized flow but also medium-sized flow. They have proposed an SRAM/DRAM model. The SRAM is used to filter out the small flow and passes the medium and large-sized flow to DRAM. It handles the elephant flow and mice flow by dividing the redundant paths into high throughput and low latency paths. It schedules the mice flow to the low latency paths and the elephant flow from the high latency paths. There are some efficient approaches for traffic management are also given by researchers Tiwari et al. (2019) and Khan et al. (2018), mentioning them for completeness.

Flow Characterization

The characterization of the flow is important for traffic engineering purposes. Various researchers have characterized the heavy hitter flows with different classification schemes, such as size, duration, rate, and burst. Based on this, the flows are categorized as follows.

Elephant Flow and Mice Flow (Lan & Heidemann, 2006)

The flow has been categorized based on the flow size into two categories Elephant and Mice. The elephant flow is a large continuous flow measured on a network connection. This flow do not occur frequently but it occupies a large portion of network bandwidth for longer durations. On the other side, the mice flow is a short flow (computed as total number of bytes) on the network connection.

Tortoise and Dragonfly flow (Brownlee & Claffy, 2002)

The flow can also be classified based on the duration of the flow as described in (Brownlee & Claffy, 2002). The work identifies that nearly 45% of the flows have a traffic duration of less than 2 seconds. The flow with a higher duration (usually, greater than 15 minutes) are classified as the tortoise flows and the remaining flows are classified as the dragonfly flow. Generally, less than 2% of the flows are tortoise flows in the network and these flows carry more than 50% of the total data on a link.

Cheetah and Rat flow (Lan & Heidemann, 2006)

The flow can also be classified based on the rate of the flow. The flow with a higher data rate from a certain threshold value are categorized as cheetah flow, whereas the flow lower than the threshold is classified as rat flow. For example, the cheetah flow can be defined as those flows that have rate greater than 100 KB per second and the remaining traffic which have lesser rate, can be defined as the rat flow (Lan & Heidemann, 2003).

PROPOSED METHODOLOGY

In the proposed mechanism, the Flow Monitor module listens to OpenFlow messages PacketIn and FlowRemoved messages. When a PacketIn message is received, then the proposed methodology extracts the match from it and uses the host IP to create a list of hosts. Using the OFSwitch instance, a list of all switches and switch port mapping is created. Each flow entry has two timers (Hard timeout and Idle timeout) and the expiration of any of them leads to the removal of the flow. Whenever a flow is removed a FlowRemoved message is received by the controller from the switch. When a timeout occurs the flow entry expires. It informs several properties of the expired entry to the controller, such as the duration of the flow, the amount of traffic matched against it, and the match of the flow (containing IP, protocol, etc.). Using this information, the data transfer rate of the flow and the

duration of the flow is computed. This information is further processed to identify the elephant flow along with cheetah flow and mice flow.

At the controller, the flow monitor module is running which has the listener for the PacketIn and the FlowRemoved message as shown in Figure 1. The flow monitor module is running a schedule for a fixed period, which runs the procedures of elephant flow and cheetah flow detection. After receiving a flow, the information is added to the list for each type of heavy hitter flow.

Algorithm 1 describes the processing of the PacketIn and FlowRemoved messages. When the controller receives a PacketIn message, the message is processed to create a host-list and switch-list. The Algorithm for processing these lists is given in Algorithm 2, which further calls two algorithms (algorithm 3 and algorithm 4) for ElephantFlow and CheetahFlow detection.

```
Algorithm 1. List Generation
Input:
P is the packet received at the controller, swID is the ID of the
sender switch
Output:
List Hlist, Slist, eflist and cflist
1.        Read asynchronous method received at controller as
Packet P.
2.        Read the sender switch ID as swID.
3.        Hlist¬NULL
4.        Slist¬NULL
5.        If p.type=PacketIn then
a.        Match¬p.match
b.        Insert match.SourceIP into Hlist
c.        Insert march.DestinationIP into Hlist
d.        Insert swlID into Slist
e.        Slist.port¬swID.activeport
6.        If p.type=FlowRemoved then
a.        match¬p.match
b.        bc¬p.getByteCount
c.        Dur¬p.Duration
d.        If p.reason=idletimeout then
i.        Dur¬ -idleTimeout
ii.        Rate¬bc/dur
e.        list1.append(match,bc)
f.        list2.append(match,rate)
g.        eflist¬list1
h.        cflist¬list2
```

The detection procedures process this list and extract specific information (byte count or duration) for creating an array and process that array to find the flows satisfying the threshold criteria.

The heuristics used for the threshold (***Th)*** computation for the flow detection is as follows (Lan & Heidemann, 2006).

1. Elephant Flow

$$\boldsymbol{Th}_{elephantflow} = \left(mean\left(SIZE\right) + 3{*}std\left(SIZE\right)\right) \tag{1}$$

2. Cheetah Flow

$$\boldsymbol{Th}_{Cheetahflow} = \left(mean\left(RATE\right) + 3*std\left(RATE\right)\right) \tag{2}$$

Here, the mean is calculated as the average for all flows and the *std* is standard deviation from the mean. The *SIZE, RATE*, and *DUR* are the arrays of the values for all flows.

```
Algorithm 2. Flow Characterization
Input: eflist,cflist
Output: ElephantFlowList,CheetahFlowList
1.          For i=1 to n do
a.          arr1[i]¬eflist.bc
b.          arr2[i]¬cflist.rate
2.          ElephantFlowRate¬ElephantFlowDetector(arr1)
3.          CheetahFlowRate¬CheetahFlowDetector(arr2)
```

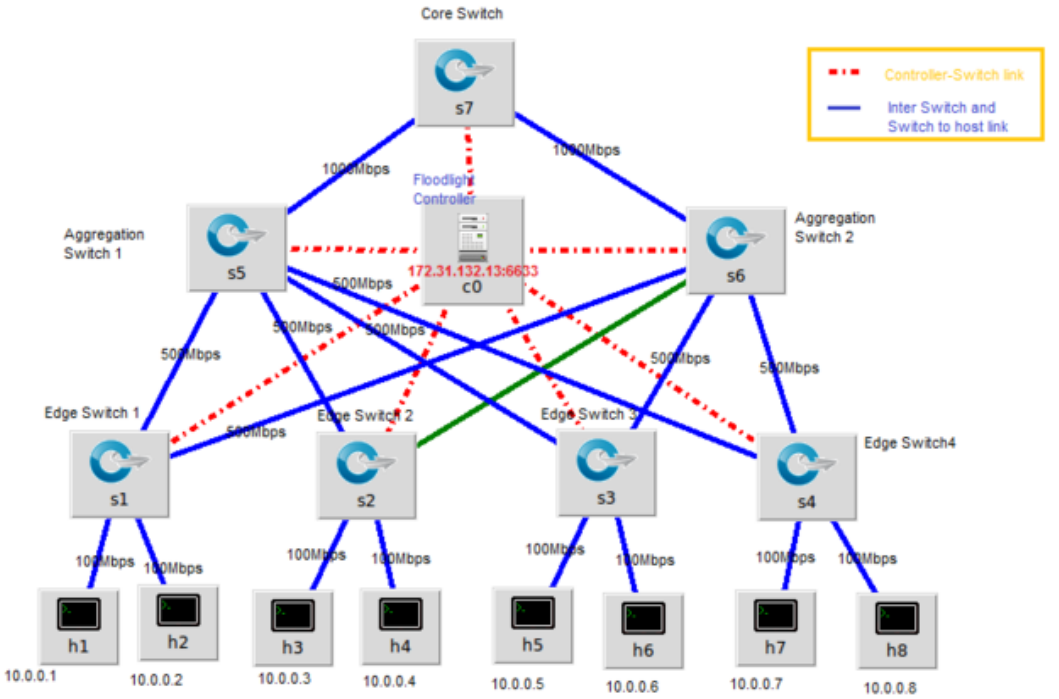
Algorithm 3 and algorithm 4 are used to detect the elephant flow and cheetah flow respectively. When the byte count is greater than the threshold value then, the flow is detected as the elephant flow as illustrated in algorithm 3. Similarly, the traffic is detected as the cheetah flow if the traffic rate of the flow is greater than the threshold.

```
Algorithm 3. Elephant Flow Detector
Input: arr1 Array of Byte count of every flow
Output: ElephantFlowList
1.          ElephantFlowList¬NULL
2.          Mean¬mean(arr1)
3.          Std¬std(arr1)
4.          Cond¬mean+3*std
5.          for i=0 to n do
a.          if arr1[i]>=cond then
b.          insert i into ElephantFlowList
6.          Return ElephantFlowList
Algorithm 4. Cheetah Flow Detector
Input: arr2 Array of rate of every flow
Output: CheetahFlowList
1.          CheetahFlowList ¬NULL
2.          Mean¬mean(arr2)
3.          Std¬std(arr2)
4.          Cond¬mean+3*std
5.          for i=0 to n do
a.          if arr2[i]>=cond then
b.          insert i into CheetahFlowList
6.          Return CheetahFlowList
```

EXPERIMENTATION AND RESULTS

To perform experimentation, the DC network is implemented on Mininet (Lantz et al., 2010). This network is a collection of network resources such as routing, load balancing, traffic analysis, etc. The Mininet tool is widely used to create a realistic virtual network that executes kernel, switches, and application codes on a single machine. Therefore, Mininet is used to develop and experiment with the SDN. The Scapy scripts (Gedia & Perigo et al., 2018) are used to generate random traffic to simulate the network traffic. The DC Network Fat-tree topology is created as shown in Figure 1. The flow Monitor module has been implemented in the floodlight controller, which monitors the flow for the size, rate, and duration.

Figure 1. Testbed topology to simulate the data center network implemented as Fat Tree topology. The topology is having one core layer switch, 2-aggregation layer switch, and 4 edge switches. The controller is connected to all the switches



RESULTS

The floodlight controller has been appended with a module, flow monitor which is used to capture the flow information received at the controller. The result has been calculated with the varying scenario, and are shown with 100 flows generated in real-time using Scapy by applying the logical scripts to simulate the data center traffic. The simulation results are shown in Figure 2, Figure 3, and Figure 4. The elephant flow threshold has been evaluated as 79000 KB by using equation 1 on the captured traffic data. With this threshold value, the 5% flows are detected as the elephant flow as shown in Figure 2. Table 2 gives the list of elephant flow detected during the simulation. As shown in Figure 3, the total traffic generated due to elephant flow constitutes 68% of the total traffic generated.

Table 2. Flows categorized as Elephant Flow

Flow ID	Source	Destination	Flow Size(KB)	Duration(ms)
47	10.0.0.1	10.0.0.4	90000	10
49	10.0.0.3	10.0.0.7	90000	7.8
76	10.0.0.3	10.0.0.8	124304	12
94	10.0.0.4	10.0.0.6	93644	8.2
98	10.0.0.1	10.0.0.8	122289	10.8

Figure 4 demonstrates the percentage utilization of the network link. The bandwidth for the link is 100Mbps. It is visible that the elephant flows consume above 90% of the bandwidth and cause congestion in the network.

Table 3 provides the Flow ID of those flows that are characterized as Cheetah flow. The results related to cheetah flow detection are shown in Figure 5. The threshold value for cheetah flow is computed as 9.7MBPS using equation 2 on the captured traffic data. With this threshold value, a total of 6% of flows are detected as the cheetah flow in the network. It is shown that all the elephant flows are included as the cheetah flow. Therefore, it can be concluded that the detection of cheetah flow is more useful as it also considers the rate of the flow.

CONCLUSION AND FUTURE WORK

In this paper, a method is proposed to detect the elephant flow and cheetah flow in a simulated DC network. The proposed method consumes lesser resources and uses an adaptive threshold based on the probability distribution. The advantage of the proposed method is that the module is implemented directly as a module in the controller and it relies only on the existing message. Thus, the method does not generate any additional network traffic. In the proposed work, the static ways are utilized for statistics collection and analysis of the traffic. The work does not consider the active statistics for processing. In future works, it would be attempted to collect the network statistics with a pull-based mechanism to minimize the network overhead by selectively choosing a switch.

The limitation of the method is that it lacks real-time statistics since it depends on the asynchronous method generated. The other limitation is that for initial flow it will detect non-elephant flow as elephant until the threshold has been established.

The future scope of this research is to add the windowed adaptive threshold calculation as well as time-based querying to avoid delays in case of only mice flows are present. The statistics can also be used to train a deep learning model that can also provide a prediction for elephant flows.

CONFLICT OF INTEREST

The authors of this publication declare there is no conflict of interest.

Figure 2. The Graph shows the Size of Flow for flow ID. The threshold value of 79000KB. The flow size above is considered an elephant flow

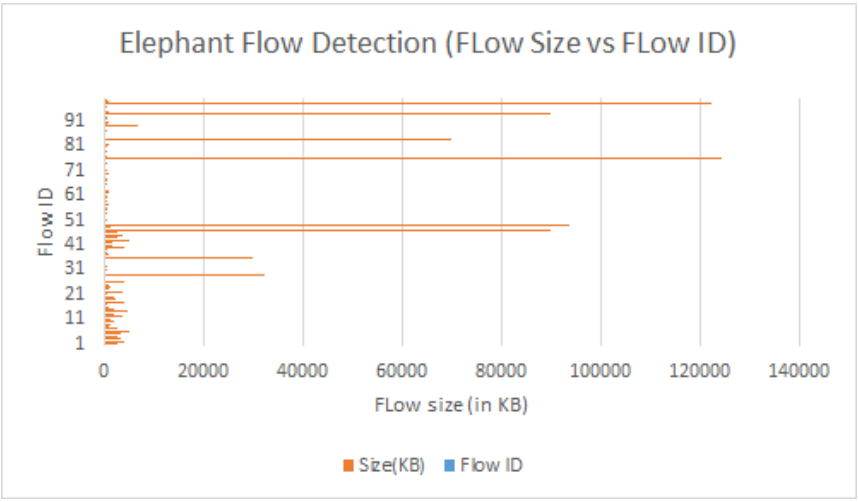


Figure 3. The Pie chart shows accumulated traffic consumed by elephant flow vs normal flow. The graph clearly shows that only 5% of flows constitute 68% of the total traffic.

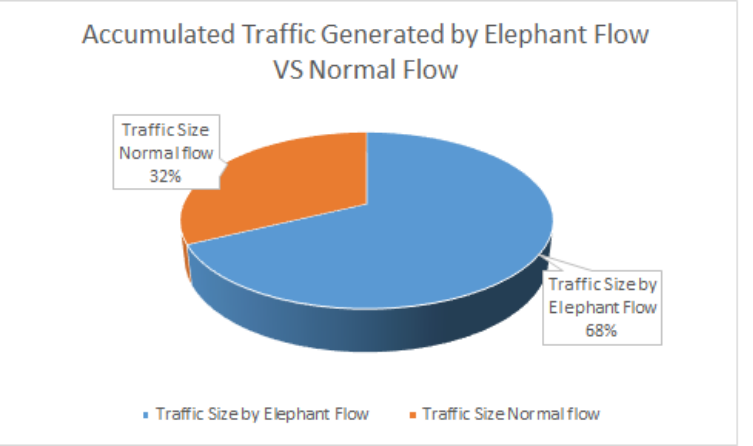


Figure 4. Percentage utilization of the link

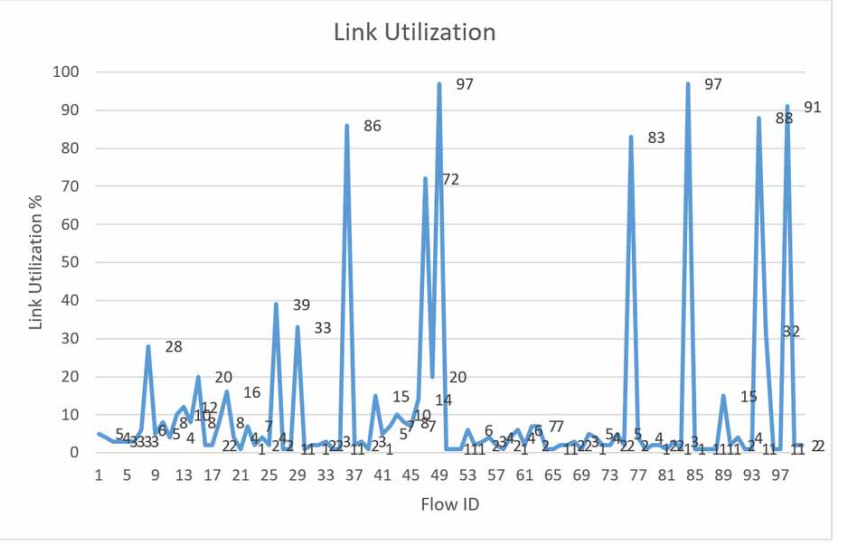
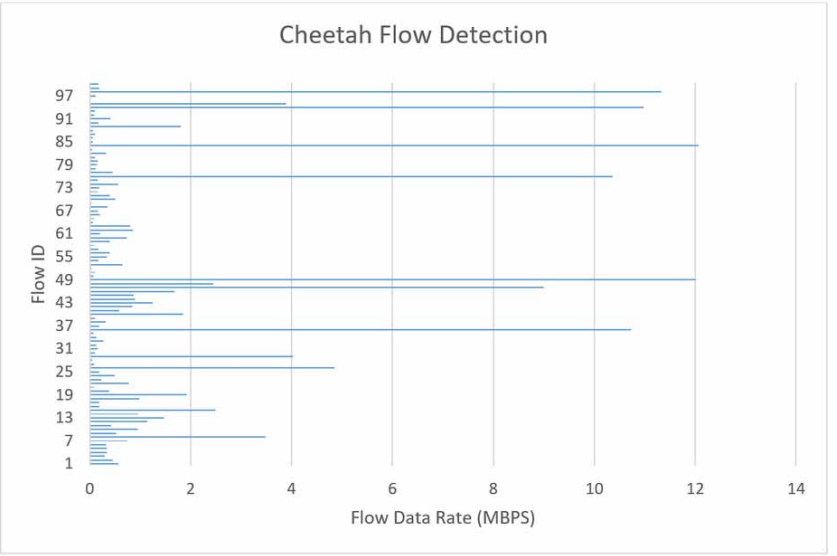


Table 3. Flows categorized as Cheetah Flow

Flow ID	Source	Destination	Flow Size(KB)	Rate(MBPS)
36	10.0.0.1	10.0.0.5	3045	12.07
49	10.0.0.3	10.0.0.7	93644	12
76	10.0.0.3	10.0.0.8	124304	10.35
84	10.0.0.1	10.0.0.4	70000	12.06
94	10.0.0.3	10.0.0.7	90000	10.98
98	10.0.0.1	10.0.0.8	122289	11.32

Figure 5. The Graph shows the Flow rate in MBPS for flow ID. The threshold value of 9.9MBPS. The flow data rate above this is considered Cheetah flow.



FUNDING AGENCY

This research work was supported by University of Petroleum and Energy Studies.

REFERENCES

Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., & Vahdat, A. (2010, April). Hedera: dynamic flow scheduling for data center networks. In NSDI (Vol. 10, No. 8, pp. 89-92). Academic Press.

Benson, T., Anand, A., Akella, A., & Zhang, M. (2011, December). MicroTE: Fine grained traffic engineering for data centers. In *Proceedings of the seventh conference on emerging networking experiments and technologies* (pp. 1-12). doi:10.1145/2079296.2079304

Brownlee, N., & Claffy, K. C. (2002). Understanding Internet traffic streams: Dragonflies and tortoises. *IEEE Communications Magazine*, *40*(10), 110–117. doi:10.1109/MCOM.2002.1039865

Chowdhury, S. R., Bari, M. F., Ahmed, R., & Boutaba, R. (2014, May). Payless: A low cost network monitoring framework for software defined networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)* (pp. 1-9). IEEE. doi:10.1109/NOMS.2014.6838227

Curtis, A. R., Kim, W., & Yalagandula, P. (2011, April). Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In 2011 Proceedings IEEE INFOCOM (pp. 1629-1637). IEEE.

Gedia, D., & Perigo, L. (2018, November). Performance evaluation of SDN-VNF in virtual machine and container. In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* (pp. 1-7). IEEE. doi:10.1109/NFV-SDN.2018.8725805

Hamdan, M., Mohammed, B., Humayun, U., Abdelaziz, A., Khan, S., Ali, M. A., Imran, M., & Marsono, M. N. (2020). Flow-aware elephant flow detection for software-defined networks. *IEEE Access: Practical Innovations, Open Solutions*, *8*, 72585–72597. doi:10.1109/ACCESS.2020.2987977

Ian, A., Ahyoung, L., Pu, W., Min, L., & Wu, C. (2014). A roadmap for traffic engineering in software defined networks. *Computer Networks*, *71*, 1–30. doi:10.1016/j.comnet.2014.06.002

Index, C. G. C. (2016). *Cisco Global Cloud Index: Forecast and Methodology, 2015–2020.* Cisco Public White Paper.

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., & Vahdat, A. (2013). B4: Experience with a globally-deployed software defined WAN. *Computer Communication Review*, *43*(4), 3–14. doi:10.1145/2534169.2486019

Kandula, S., Sengupta, S., Greenberg, A., Patel, P., & Chaiken, R. (2009, November). The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement* (pp. 202-208). doi:10.1145/1644893.1644918

Khan, E., Garg, D., Tiwari, R., & Upadhyay, S. (2018, February). Automated toll tax collection system using cloud database. In *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)* (pp. 1-5). IEEE. doi:10.1109/IoT-SIU.2018.8519929

Kumar, S., & Tiwari, R. (2021). Dynamic popularity window and distance-based efficient caching for fast content delivery applications in CCN. *Engineering Science and Technology, an International Journal, 24*(3), 829-837.

Kumar, S., Tiwari, R., & Hong, W. C. (2021). QoS Improvement Using In-Network Caching Based on Clustering and Popularity Heuristics in CCN. *Sensors (Basel)*, *21*(21), 7204. doi:10.3390/s21217204 PMID:34770508

Lan, K. C., & Heidemann, J. (2003). *On the correlation of internet flow characteristics*. Technical Report ISI-TR-574, USC/ISI.

Lan, K. C., & Heidemann, J. (2006). A measurement study of correlations of internet flow characteristics. *Computer Networks*, *50*(1), 46–62. doi:10.1016/j.comnet.2005.02.008

Lantz, B., Heller, B., & McKeown, N. (2010, October). A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (pp. 1-6). doi:10.1145/1868447.1868466

Lin, C. Y., Chen, C., Chang, J. W., & Chu, Y. H. (2014, December). Elephant flow detection in datacenters using openflow-based hierarchical statistics pulling. In *2014 IEEE Global Communications Conference* (pp. 2264-2269). IEEE. doi:10.1109/GLOCOM.2014.7037145

Luong, D. H., Outtagarts, A., & Hebbar, A. (2016, June). Traffic monitoring in software defined networks using opendaylight controller. In *International Conference on Mobile, Secure, and Programmable Networking* (pp. 38-48). Springer. doi:10.1007/978-3-319-50463-6_4

Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys and Tutorials*, *16*(3), 1617–1634. doi:10.1109/SURV.2014.012214.00180

Queiroz, W., Capretz, M. A., & Dantas, M. (2019). An approach for SDN traffic monitoring based on big data techniques. *Journal of Network and Computer Applications*, *131*, 28–39. doi:10.1016/j.jnca.2019.01.016

Sahu, H., & Hungyo, M. (2018). Introduction to SDN and NFV. In *Innovations in Software-Defined Networking and Network Functions Virtualization* (pp. 1–25). IGI Global. doi:10.4018/978-1-5225-3640-6.ch001

Sarvotham, S., Riedi, R., & Baraniuk, R. (2001, November). Connection-level analysis and modeling of network traffic. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement* (pp. 99-103). doi:10.1145/505202.505215

Singh, U., Vankhede, V., Maheshwari, S., Kumar, D., & Solanki, N. (2019, August). Review of Software Defined Networking: Applications, Challenges and Advantages. In *International Conference on Inventive Computation Technologies* (pp. 815-826). Springer.

Tiwari, R., Sharma, H. K., Upadhyay, S., Sachan, S., & Sharma, A. (2019). Automated parking system-cloud and IoT based technique. *International Journal of Engineering and Advanced Technology*, *8*(4C), 116–123.

Tootoonchian, A., Ghobadi, M., & Ganjali, Y. (2010, April). OpenTM: traffic matrix estimator for OpenFlow networks. In *International Conference on Passive and Active Network Measurement* (pp. 201-210). Springer. doi:10.1007/978-3-642-12334-4_21

Wang, W., Sun, Y., Zheng, K., Kaafar, M. A., Li, D., & Li, Z. (2014, October). Freeway: Adaptively isolating the elephant and mice flows on different transmission paths. In *2014 IEEE 22nd international conference on network protocols* (pp. 362-367). IEEE.

Xing, C., Ding, K., Hu, C., & Chen, M. (2016). Sample and fetch-based large flow detection mechanism in software defined networks. *IEEE Communications Letters*, *20*(9), 1764–1767. doi:10.1109/LCOMM.2016.2585480

Yu, C., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G., & Madhyastha, H. V. (2013, March). Flowsense: Monitoring network utilization with zero measurement cost. In *International Conference on Passive and Active Network Measurement* (pp. 31-41). Springer. doi:10.1007/978-3-642-36516-4_4

Zaw, H. T., & Maw, A. (2019). Traffic management with elephant flow detection in software defined networks (SDN). *Iranian Journal of Electrical and Computer Engineering*, *9*(4), 3203. doi:10.11591/ijece.v9i4.pp3203-3211